AF

DOCKET NO. 00-LJ-217 (STMI01-00217) **PATENT**
Customer No. 30425

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In application of	:	Faraydon O. Karim, et al.
U.S. Serial No.	:	09/917,290
Filed	:	July 27, 2001
For	:	NOVEL FETCH BRANCH ARCHITECTURE FOR REDUCING BRANCH PENALTY WITHOUT BRANCH PREDICTION
Group No.	:	2183
Examiner	:	Aimee J. Li

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

Sir:

The undersigned hereby certifies that the following documents:

1. Appellant's Brief on Appeal;
2. Check in the amount of $500.00 for the Appeal Brief filing fee; and
3. A postcard receipt

relating to the above application, were deposited as "First Class Mail" with the United States Postal Service, addressed to MAIL STOP APPEAL BRIEF-PATENTS, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on July 5, 2005.

Date: 7-5-05

Mailer

Date: 7-5-05

Daniel E. Venglarik
Reg. No. 39,409

P.O. Box 802432
Dallas, Texas 75380
Phone: (972) 628-3600
Fax: (972) 628-3616
E-mail: *dvenglarik@davismunck.com*

DOCKET NO. 00-LJ-217 (STMI01-00217) **PATENT**
Customer No. 30425

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	:	Faraydon O. Karim, et al.
Serial No.	:	09/917,290
Filed	:	July 27, 2001
For	:	NOVEL FETCH BRANCH ARCHITECTURE FOR REDUCING BRANCH PENALTY WITHOUT BRANCH PREDICTION
Group No.	:	2183
Examiner	:	Aimee J. Li

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

APPELLANTS' BRIEF ON APPEAL

This Brief is submitted on behalf of Appellant for the application identified above. A check for the $500.00 fee for filing a brief on appeal is enclosed. Please charge any additional necessary fees to Deposit Account No. 50-0208.

07/12/2005 MBERHE 00000010 09917290
01 FC:1402 500.00 OP

REAL PARTY IN INTEREST

The real party in interest for this appeal is the assignee of the application, STMICRO-ELECTRONICS, INC. (f/k/a SGS-THOMSON MICROELECTRONICS, INC.).

RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences related to the present application which are currently pending.

STATUS OF CLAIMS

Claims 1–20 are pending in the present application. Claims 1–20 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,127,091 to *Boufarah et al* in view of U.S. Patent No. 6,289,445 to *Ekner*. The rejection of pending claims 1–20 is appealed.

STATUS OF AMENDMENTS

No amendments to the claims were submitted following the final Office Action mailed January 4, 2005.

SUMMARY OF THE CLAIMED SUBJECT MATTER

The claimed invention relates to a branch architecture limiting potential branch penalty by pre-fetching both the next sequential instruction (following the last fetched instruction) and the branch target instruction during the two cycles following dispatch of the branch instruction to the execution unit. During program execution, a change in program flow due to branching instructions (executing instructions from a different memory address rather than simply the next sequential

address) can result in delay of program execution. Specification, page 2, lines 4–16; U.S. Patent Application Publication No. 2003/0023838 ("Publication"), ¶ [0004]. While most contemporary processors attempt to avoid such delay by implementing some form of branch prediction, misprediction can result in a significant performance penalty, particularly where a large number of clock cycles separates the fetch and branch units. Specification, page 2, line 17 through page 4, line 5; Publication, ¶¶ [0005]–[0008]. In additional, aggressive pipelining and branch prediction can increase design complexity, etc. Specification, page 4, lines 6–17; Publication, ¶ [0009].

In the present invention, both design complexity and branch prediction error performance penalty are kept minimal by implementing a merged fetch-branch unit 103 operating in parallel with the decode unit 102, and fetching both the next sequential instruction (S) following the branch instruction (B) and the branch target instruction (T) at the branch target address during the two cycles following detection of the branch instruction (B):

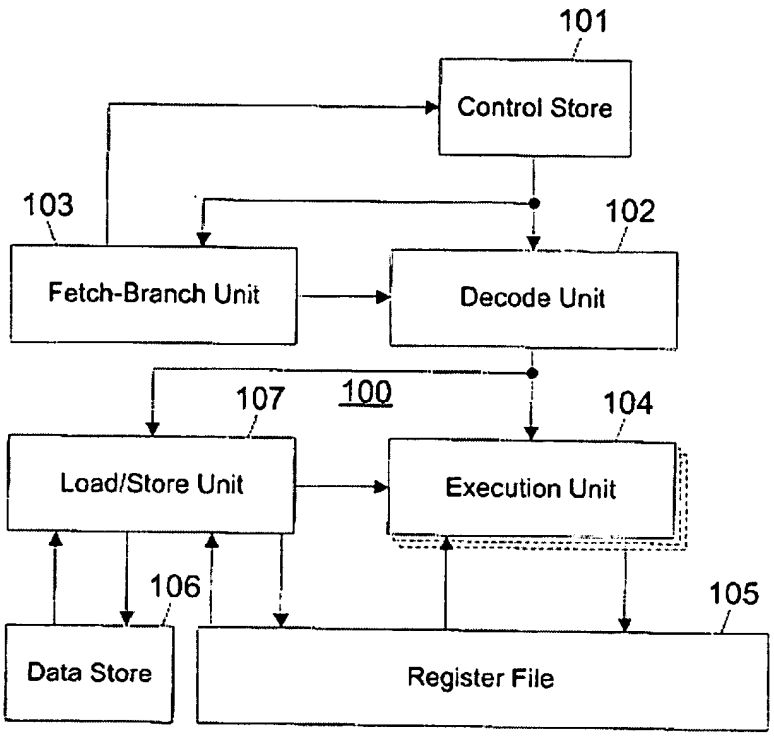

Figure 1

Specification, Figure 1, page 9, line 11 through page 15, line 7; Publication, Figure 1, ¶¶ [0019]–[0041]. After two clock cycles the branch instructions is resolved in the combined fetch-branch unit 103; if the branch is taken, the sequential instruction S is discarded; if the branch is not taken, the target instruction T is discarded. Specification, page 15, lines 8–10; Publication, ¶ [0042]. In either case, a fixed, maximum performance penalty of one clock cycle (one instruction) is incurred.

In this manner, the branch penalty is fixed at one cycle. Unused (sequential or target)

instructions are dropped upon resolution of the branch instruction. Specification, page 15, lines 8–10; Publication, ¶ [0042]. Instructions are marked in the fetch-branch unit as "regular," "branch," "sequential," or "target" to facilitate such purging of unused instructions. Specification, page 11, line 22 through page 12, line 16; Publication, ¶¶ [0024]–[0028].

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1–20 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,127,091 to *Boufarah et al* in view of U.S. Patent No. 6,289,445 to *Ekner*.

ARGUMENT

A. Claims 1–20

These claims are properly grouped together and considered separately because a favorable decision with respect to these claims may obviate the need for separate consideration of the remaining claims (even though unpatentability of these claims is NOT determinative of patentability of the other claims discussed below due to the distinguishing claim feature(s) identified below).

In *ex parte* examination of patent applications, the Patent Office bears the burden of establishing a *prima facie* case of obviousness. MPEP § 2142, p. 2100-128 (8th ed. rev. 2 May 2004). Absent such a *prima facie* case, the applicant is under no obligation to produce evidence of nonobviousness. *Id.*

To establish a *prima facie* case of obviousness, three basic criteria must be met: First, there must be some suggestion or motivation, either in the references themselves or in the knowledge

generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. *Id.*

Independent claims 1, 8 and 15 each recite a fetch-branch unit and a decode unit. In addition, claim 8 further recites at least one execution unit. Such a combination of features is not found in the cited references. *Boufarah et al* depicts an instruction fetching and branch processing unit 14, a fixed point processing unit 20, and other processing units 22:

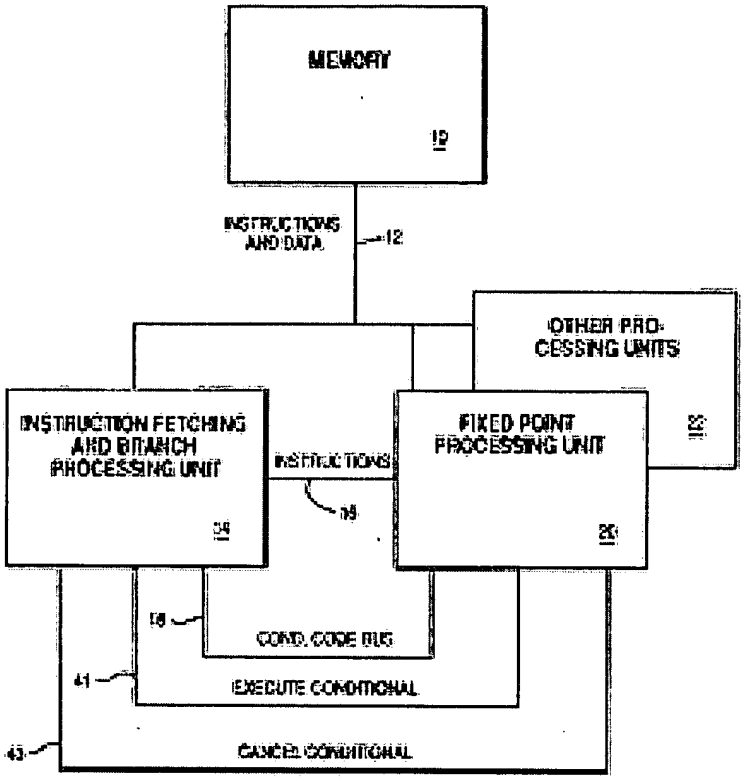

FIG. 1

Boufarah et al, Figure 1. However, *Boufarah et al* does not depict a separate decode unit, but instead teaches that instructions are decoded in the fixed point execution unit 20 (or 22):

> Meanwhile in cycle 4 the fixed point processing unit 20 decodes the instruction X2 and executes instruction X1.

Boufarah et al, column 4, lines 9–10. Thus *Boufarah et al* does not disclose "a decode unit," separate and distinct from execution unit(s). Similarly, *Ekner* discloses a fetch unit 302 and a decode-branch unit 304, rather than a fetch-branch unit and decode unit.

The Advisory Action mailed April 11, 2005 states:

> Applicant argues in essence on pages 11–12 "Thus Boufarah et al does not disclose 'a decode unit,' separate and distinct from execution unit(s)." This has not been found persuasive. The claim language relied upon merely states "at least one execution unit . . . a decode unit . . ." but does not state that they must be separate and distinct. Since Boufarah's fixed point execution unit decodes and executes instructions it is a decode unit and at least one execution unit. There is nothing in the claim language that suggests the decode unit must [be] separate and distinct from the execution unit(s) and cannot be the same device, as is taught by Boufarah.

Paper No. 20050404, page 2. However, the claims separately recite the decode and at least one execution units as separate elements. Unlike recitation of "means for" elements, a claim that specifically calls out separate physical elements cannot be interpreted as satisfied by a single element.

Regardless, independent claims 1, 8 and 15 also each recite that the fetch-branch unit and decode units operate in parallel, the fetch-branch unit and the decode unit (a) both receiving the same instruction(s) during a given cycle and (b) having the sequential or target instructions retrieved thereto concurrently during one of the following two cycles. Such a feature is not found in the cited references. As conceded in the final rejection, such a feature is not found in *Boufarah et al.* Contrary to the assertion within the final rejection, such a feature is also not found in *Ekner*. The cited portion of *Ekner* depicts a unit 304 decoding instructions and detecting branch instructions operating in series with a fetch unit 302 within the execution pipeline:

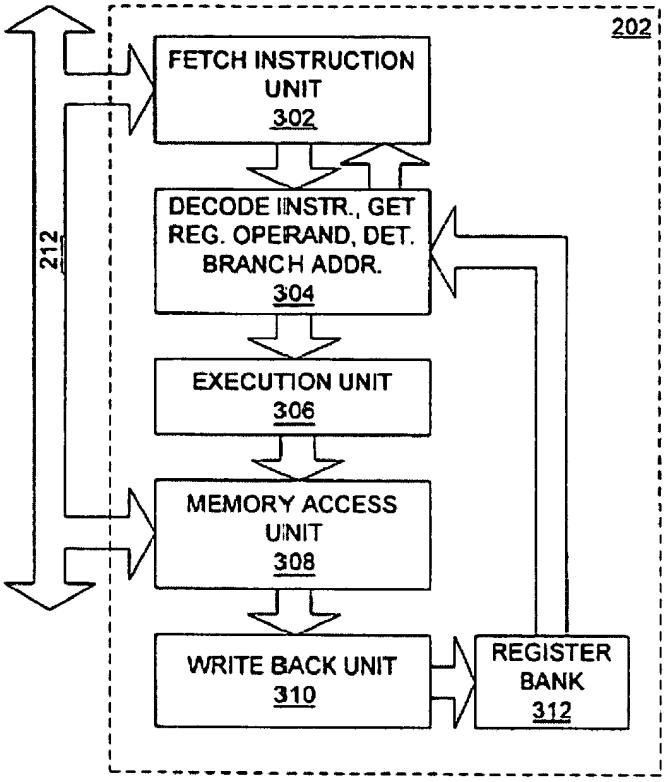

FIG. 2

Ekner, Figure 2, column 5, lines 52–65. *Ekner* teaches a fetch unit and a decode-branch unit, NOT a fetch-branch unit and a decode unit as recited in the claims. Furthermore, instructions are NOT received concurrently at the fetch unit 302 and the decode-branch unit 304 in *Ekner*, as recited in the claims. The fact that communication between the fetch unit 302 and the decode-branch unit 304 is bidirectional does NOT satisfy the claim limitations that the same instructions are received at both units during a given cycle, or that sequential or target instructions are retrieved to both units during

the same (subsequent) cycle.

In addition, *Ekner* teaches speculative execution of instructions, but does NOT teach specifically fetching instructions from both sequential and target locations for a branch instruction during the two clock cycles following detection of the branch instruction. Accordingly, the references do not teach all limitations of the claims.

B. Claims 2, 9 and 16

These claims are properly grouped together and considered separately from the other groups of claims because these claims recite a feature distinguishing the claimed invention over the prior art references cited that is not found in all of the claims within the other group(s) of claims: recite dropping either the sequential instruction or the target instruction from both the fetch-branch unit and the decode unit upon resolving the branch instruction. The claims of this group are thus separately patentable from claims within the other group(s) over the teachings of the cited prior art.

Claims 2, 9 and 16 each recite dropping either the sequential instruction or the target instruction from both the fetch-branch unit and the decode unit upon resolving the branch instruction. Such a feature is not found in the cited references. Instead, *Boufarah et al*, which fetches four instructions during each execution cycle, speculatively executes the sequential instructions, and purges those instructions from the processing unit 20 (those instructions having already been forwarded from the fetch-branch unit 14) if resolution of the branch instruction indicates that the target instructions should be executed instead. *Boufarah et al*, column 4, line 29 through column

5, line 60. To the extent the processing unit 20 is taken as satisfying the recited limitation of "a decode unit," *Boufarah et al* does not teach dropping either the sequential instructions or the target instructions from both the fetch-branch unit and the decode unit upon resolution of the branch instruction as recited in the claims. In the system disclosed by *Boufarah et al*, if the sequential instructions are to be executed, the target instructions are never forwarded to the processing unit 20 for execution, and therefore cannot be "dropped" from the processing unit 20 as well as the fetch-branch unit 14; on the other hand if the target instructions are to be executed, the sequential instructions are purged only from the processing unit 20 in response to resolution of the branch instruction, not from both fetch-branch unit 14 and processing unit 20 (those sequential instructions having previously been eliminated from fetch-branch unit 14 during transfer to the processing unit 20).

C. Claims 4–7, 11–14 and 18–20

These claims are properly grouped together and considered separately from the other groups of claims because these claims recite a feature distinguishing the claimed invention over the prior art references cited that is not found in all of the claims within the other group(s) of claims: marking instructions with various identifiers: regular, branch, sequential and/or target. The claims of this group are thus separately patentable from claims within the other group(s) over the teachings of the cited prior art.

Claims 4–7, 11–14 and 18–20 each recite marking instructions with various identifiers:

regular, branch, sequential and/or target. Such a feature is not found in the cited references. The labels "X1," "X2," "BRC" and "S1" through "S3" within *Boufarah et al* are simply references used to identify the respective types of instructions for the purposes of description. Nowhere does *Boufarah et al* teach actually marking the instructions with those identifiers within the processor described. Moreover, identifying instructions as one of the types of instructions described (regular, branch, sequential, or target) does not constitute "inherent" marking of the instructions as asserted in the Office Action, since the ability to differentiate instructions based on other factors does not constitute marking of the instructions. Regardless, the claim requires explicit marking of the instruction types, to facilitate purging of either sequential or target instructions.

CONCLUSION

None of the cited references, taken alone or in combination, depict or describe all features of the claimed invention in the appealed claims. Therefore, the rejections under 35 U.S.C. § 103 is improper. Applicant respectfully requests that the Board of Appeals reverse the decision of the Examiner below rejecting pending claims 1–20 in the application.

Respectfully submitted,

DAVIS MUNCK, P.C.

Date: 7-5-05

Daniel E. Venglarik
Registration No. 39,409

P.O. Drawer 800889
Dallas, Texas 75380
(972) 628-3621 (direct dial)
(214) 922-9221 (main number)
(214) 969-7557 (fax)
E-mail: *dvenglarik@davismunck.com*

CLAIMS APPENDIX

1. For use in a processor, a branch architecture for limiting branch penalty without branch prediction comprising:

a fetch-branch unit operating in parallel with a decode unit and controlling retrieval of instructions for the decode unit, both the fetch-branch unit and the decode unit receiving the same instruction(s) during a given cycle, wherein the fetch-branch unit, upon detecting a branch instruction during one cycle,

initiates retrieval to both the fetch-branch unit and the decode unit of at least one sequential instruction from a location immediately following a location of a last retrieved instruction during one of a first cycle immediately following the one cycle and a second cycle immediately following the first cycle, and

initiates retrieval to both the fetch-branch unit and the decode unit of at least one target instruction from a target location for the branch instruction during the other of the first cycle immediately following the one cycle and the second cycle immediately following the first cycle.

2. The branch architecture as set forth in Claim 1 wherein the fetch-branch unit resolves the branch instruction and, upon resolving the branch instruction, causes both the fetch-branch unit and the decode unit to drop either the at least one sequential instruction or the at least one target instruction.

3. The branch architecture as set forth in Claim 2 wherein the fetch-branch unit, upon resolving the branch instruction, initiates retrieval to both the fetch-branch unit and the decode unit of at least one instruction from a location immediately following a location of a last retrieved instruction within either the at least one sequential instruction or the at least one target instruction, depending upon whether a branch is taken.

4. The branch architecture as set forth in Claim 1 wherein the fetch-branch unit, upon detecting a branch instruction during the one cycle, marks any fetched instruction preceding the branch instruction with a regular instruction type identifier, marks the branch instruction with a branch instruction type identifier, and marks any fetched instruction succeeding the branch instruction with a sequential instruction type identifier.

5. The branch architecture as set forth in Claim 4 wherein the fetch-branch unit, upon not detecting a branch instruction during the one cycle, marks all fetched instruction(s) with the regular instruction type identifier.

6. The branch architecture as set forth in Claim 1 wherein the fetch-branch unit marks the at least one sequential instruction with a sequential instruction type identifier.

7. The branch architecture as set forth in Claim 1 wherein the fetch-branch unit marks the at least one target instruction with a target instruction type identifier.

8. A processor comprising:

at least one execution unit;

a decode unit; and

a branch architecture for limiting branch penalty without branch prediction comprising:

a fetch-branch unit operating in parallel with the decode unit and controlling retrieval of instructions for the decode unit, both the fetch-branch unit and the decode unit receiving the same instruction(s) during a given cycle, wherein the fetch-branch unit, upon detecting a branch instruction during one cycle,

initiates retrieval to both the fetch-branch unit and the decode unit of at least one sequential instruction from a location immediately following a location of a last retrieved instruction during one of a first cycle immediately following the one cycle and a second cycle immediately following the first cycle, and

initiates retrieval to both the fetch-branch unit and the decode unit of at least one target instruction from a target location for the branch instruction during the other of the first cycle immediately following the one cycle and the second cycle immediately following the first cycle.

9. The processor as set forth in Claim 8 wherein the fetch-branch unit resolves the branch instruction and, upon resolving the branch instruction, causes both the fetch-branch unit and the decode unit to drop either the at least one sequential instruction or the at least one target instruction.

10. The processor as set forth in Claim 9 wherein the fetch-branch unit, upon resolving the branch instruction, initiates retrieval to both the fetch-branch unit and the decode unit of at least one instruction from a location immediately following a location of a last retrieved instruction within either the at least one sequential instruction or the at least one target instruction, depending upon whether a branch is taken.

11. The processor as set forth in Claim 9 wherein the fetch-branch unit, upon detecting a branch instruction during the one cycle, marks any fetched instruction preceding the branch instruction with a regular instruction type identifier, marks the branch instruction with a branch instruction type identifier, and marks any fetched instruction succeeding the branch instruction with a sequential instruction type identifier.

12. The processor as set forth in Claim 11 wherein the fetch-branch unit, upon not detecting a branch instruction during the one cycle, marks all fetched instruction(s) with the regular instruction type identifier.

13. The processor as set forth in Claim 8 wherein the fetch-branch unit marks the at least one sequential instruction with a sequential instruction type identifier.

14. The processor as set forth in Claim 8 wherein the fetch-branch unit marks the at least one target instruction with a target instruction type identifier.

15. For use in a processor, a method of processing branch instructions without branch prediction comprising:

operating a fetch-branch unit in parallel with a decode unit to control retrieval of instructions for the decode unit, wherein the same instruction(s) are retrieved to both the fetch-branch unit and the decode unit during a given cycle; and

upon detecting a branch instruction during one cycle,

initiating retrieval to both the fetch-branch unit and the decode unit of at least one sequential instruction from a location immediately following a location of a last retrieved instruction during one of a first cycle immediately following the one cycle and a second cycle immediately following the first cycle, and

initiating retrieval to both the fetch-branch unit and the decode unit of at least one target instruction from a target location for the branch instruction during the other of the first cycle immediately following the one cycle and the second cycle immediately following the first cycle.

16. The method as set forth in Claim 15 further comprising:

resolving the branch instruction; and

upon resolving the branch instruction, causing both the fetch-branch unit and the decode unit to drop either the at least one sequential instruction or the at least one target instruction.

17. The method as set forth in Claim 16 further comprising:

upon resolving the branch instruction, retrieving, to both the fetch-branch unit and the decode unit, at least one instruction from a location immediately following a location of a last retrieved instruction within either the at least one sequential instruction or the at least one target instruction, depending upon whether a branch is taken.

18. The method as set forth in Claim 15 further comprising:

upon detecting a branch instruction during the one cycle,

marking any fetched instruction preceding the branch instruction with a regular instruction type identifier,

marking the branch instruction with a branch instruction type identifier, and

marking any fetched instruction succeeding the branch instruction with a sequential instruction type identifier.

19. The method as set forth in Claim 18 further comprising:

upon not detecting a branch instruction during the one cycle, marking all fetched instruction with the regular instruction type identifier.

20. The method as set forth in Claim 15 further comprising:

marking the at least one sequential instruction with a sequential instruction type identifier; and

marking the at least one target instruction with a target instruction type identifier.

EVIDENCE APPENDIX

Not applicable.

RELATED PROCEEDINGS APPENDIX

Not applicable.